Comparing Recurrent Neural Networks & Pretrained Language Models on Multiclass Topic Classification

Frieso Turkstra
s3788946

Student 1
s123456

Student 2
s123456

Abstract

In this study, we tackle the task of multiclass topic classification for online reviews of six different products. We first implement a baseline system by experimenting with various LSTM networks. We experimented with both different architectures and different hyperparameters. Our best LSTM network employed a bi-directional architecture and achieved 89.8% accuracy on the test set. Then, eight different PLMs were tested and the best one was chosen for further fine-tuning. The best model was BERT-cased which, after hyperparameter fine-tuning, achieved an accuracy of 95.2% on the test set. This confirmed our hypothesis that PLMs would outperform our LSTM baseline system on the task of multiclass topic classification.

1 Introduction

Topic classification is the process of automatically labelling and categorizing of documents. This is an important issue in a wide variety of fields, because it can help filter large bodies of unstructured textual data, as well as improve the personalization of online services. In this paper, we tackle the challenge of classifying online reviews into different categories. The reviews are split based on six different types of products. Therefore, the goal is matching each review to its corresponding product type. In order to achieve this, we first created a baseline system. More specificially, we experimented on several LSTM models by adding pre-trained embeddings, adjusting the number of LSTM layers, adding various degrees of dropout, trying different opimizers, and changing the directionality of the model. Then, we used the score of the best LSTM network for comparison against the scores of pretrained language models (PLM). We experimented with fine-tuning a number of PLMs and selected the model most suited to our task. Lastly, we fine-tuned the hyperparameters for the highest scoring PLM. The resulting best model is the cased version of BERT with an accuracy of 95.2% on the test set.

2 Related Work

When it comes to topic classification, state of the art results are usually achieved by fine-tuning pretrained language models on the task. For example, in a shared task to automatically categorize Covid-19 related articles, it is reported that the top five best scorers all include fine-tuning of a PLM in their approach (Chen et al., 2022). Similarly, on the task of classifying reviews on electric vehicle charging charging stations, fine-tuning BERT resulted in an accuracy of 91.6%, whereas an LSTM model scored 90.3% (Ha et al., 2021). The current task is quite similar to the ones presented previously, so it follows that we may see comparable scores. However, since the topics and datasets differ, there is potential for divergent results. In any case, we do expect the PLMs to outperform our best LSTM network.

3 Data

The data for this study consists of 6,000 reviews divided over six categories: music, DVD, camera, health, software, and books. All reviews are written in English and manually annotated for one of the six categories. The data was randomly split into a training, development and test set at an 80-10-10 ratio. The exact distribution of the topics in each data set is presented in Table 1 (van der Holt et al., 2023). The data set has been tokenized.

	Train	Development	Test
Music	807	109	111
DVD	788	119	105
Camera	793	96	99
Health	809	95	82
Software	819	80	95
Books	784	101	108
Total	4,800	600	600

Table 1: Distribution of topics in the train, development and test set. Note. From "Multi-Class Topic Classification of Reviews," by M. van der Holt et al., p. 1.

4 Method/Approach

We first created a baseline system by finding the optimal architecture and parameters for different LSTM networks. In the second round of experiments, we use PLMs. Multiple models were compared and the best model was fine-tuned to further push performance. This fine-tuned model is the system we used to get our final performance scores.

4.1 LSTM

A Long Short-Term Memory network (LSTM) is a type of recurrent neural network that is designed to capture long-term dependencies. A connected LSTM cell receives not only an input value but also the state and output of the previous cell. The previous cell state bears the most importance and is modulated by three gates. The first gate determines how much information we want to remove from the cell state and is accordingly called the "forget" gate. The gate consists of a sigmoid layer which determines for each value in the cell state how much it should be remembered. The second gate manages how much new information should be added to the cell state. Lastly, the third gate determines which parts of the cell state should be outputted. Thus, an LSTM network can capture long-term dependencies by carefully selecting what needs to be remembered from previous states and deciding what needs to be updated and what is relevant for future states. (Schmidhuber et al., 1997).

4.1.1 Pre-trained embeddings

Firstly, we performed experiments to see the effect of pre-trained embeddings on the model. The best model without pre-trained embeddings had an output dimension of 146 on the embedding layer, and achieved an accuracy of 84.5% on the development set over 10 epochs. For the pre-trained embeddings, we used GloVe embeddings (Pennington et al., 2014). The highest scoring model with pre-trained embeddings scored an accuracy of 90.5% over 10 epochs. This model had trainable embeddings, meaning that instead of being frozen, the weights of the pre-trained embeddings can still be updated during training. From this, we conclude trainable pre-trained embeddings are most suitable given the current task and dataset.

4.1.2 Multiple LSTM layers

Then, further experiments were done with multiple LSTM layers. To stack LSTM layers, the output size of an LSTM cell needs to match the input size of the next. This means it should not have a single output value but a series of previous cell states. All experiments were performed by running the model for 50 epochs but using an early stopping callback function with a patience value of 5 that stops the training when the validation loss does not improve for five consecutive epochs.

As default settings, we used softmax for the activation layer and categorical cross entropy as the loss function. The Adam optimizer proved superior over the stochastic gradient descent optimizer and the default batch size was 16.

The first experiments concerned the architecture of the model, i.e. how many layers and nodes should be used. We tested 1, 2, and 3 layers. The one-layer model yielded an accuracy of 88.2% whereas the two- and three-layer models yielded an accuracy of 89.5%. Further experiments were done with the two-layer model since it had the best score with better performance. The range of the number of nodes we tested started at 32 and went up till 512 with steps of 32. The model performed best with 512 nodes in each layer, resulting in 91% accuracy. A dropout layer in between the LSTM layers with a rate of 0.4 improved accuracy up to 91.5%.

For the learning rate, we tested values between 1e-5 and 1e-2 with a logarithmic sampling method. The optimal learning rate turned out to be 2.1313e-5 but did not improve accuracy. The accuracy scores obtained by the Keras tuner when finding the optimal values could not be reproduced. A normal run with the optimal values found by the Keras tuner yielded an accuracy of

only 86.3%. The random seeds were fixed during experiments so a possible explanation could be that the Keras tuner uses some other type of random initialisation of weights behind the scenes. For this reason, the last experiments on the batch size were done without the Keras tuner. For the last experiment, batch sizes of 16, 32, 64 and 128 were tried. The best batch size was 64 with an accuracy of 88.7%.

4.1.3 Bi-directional LSTM

We experimented with a bi-directional LSTM model separately. This model used softmax as the output activation function and categorical cross entropy as the loss function. Our default batch size was 32 and the default optimizer was SGD. We did these experiments by running the model for 50 epochs but using an early stopping callback function with a patience value of 5 that stops the training when the validation loss does not improve for five consecutive epochs. The (intermediate) results mentioned here are therefore not all for the same number of epochs.

First we experimented with the number of nodes in a single layer. We tried 32, 64, 128 and 256 nodes. The best results were from the model with 64 nodes with a validation accuracy of 89.7% and a validation loss of 0.338 (trained for 25 epochs). Next we experimented with two layers, both bi-directional. The combinations we tried for the two layers were 64 & 32, 64 & 64, 32 & 64 and 128 & 64. We also tried using combinations of a regular LSTM layer with a bi-directional layer. The combinations we tried here are (regular 128, bi-directional 64), (regular 64, bi-directional 128), (bi-directional 128, regular 64) and (bi-directional 64, regular 128). There was only one model that performed better than the single-layer model. This was the model where the first layer was a normal layer with 128 nodes and the second was a bi-directional layer with 64 nodes (validation accuracy of 89.8% and validation loss of 0.338, trained for 24 epochs). The difference was so small however, that for simplicity we decided to do further experiments with the single-layer model.

With this model with one bi-directional layer with 64 nodes we first tried a different optimizer and different learning rates for the two optimizers. For both the SGD optimizer (which was used in the previous experiments) and the Adam optimizer, we tried the learning rates 0.1, 0.01 and 0.001. The best model, using the Adam optimizer with a learning rate of 0.001, had a validation accuracy of 90.3% and a validation loss of 0.305 (trained for only 3 epochs), an improvement on the previous best model. Using the Adam optimizer instead of the SGD optimizer also resulted in the model needing much less epochs to train.

Next we experimented with two variations of dropout, input dropout and recurrent dropout. For both we tried dropout rates of 0.2, 0.5 and 0.8. While the best validation accuracy was reached with a recurrent dropout of 0.2 (accuracy of 91.5%), the best validation loss was achieved with an input dropout of 0.2 (loss of 0.296). Both of these models had very similar results, but the difference in loss was slightly bigger than the difference in accuracy, so we decided to take the model with an input dropout 0.2 as the best model (accuracy of 90.7%). This model needed 6 epochs to train. A combination of input dropout of 0.2 and recurrent dropout of 0.2 was also tried but did not yield better results.

Lastly, different batch sizes were tried (16, 64 and 128, 32 was the default), but this also did not improve the model. An overview of the settings of our best model can be found in Table 2.

4.2 Pre-trained language models

To further push performance, we fine-tuned several PLMs on the task. We performed initial tests on a total of eight different models, after which we experimented further with the best model. To this end, we employed the HuggingFace transformers library (Wolf et al., 2020).

4.2.1 Comparing models

For the initial experiments, we trained the different PLMs over 5 epochs with a learning rate of 5e-5 and a batch size of 8. We implemented early stopping if the loss did not decrease for 3 consecutive epochs. Additionally, the maximum sequence length was 100, with padding enabled, and the optimizer and loss functions were Adam and categorical cross entropy, respectively.

Firstly, we tried the cased and uncased versions of the base BERT model (Devlin et al., 2019), the difference being whether the model takes into account if a word contains capital letters or not. BERT is the original transformer-based model trained for masked language modelling and next sentence prediction on 3.3 billion words from Wikipedia and a book corpus. During masked language modelling, the model is fed sentences where

Main architecture	Single bi-directional layer (64 nodes) with input dropout (0.2)
Activation	Softmax
Loss function	Categorical cross entropy
Optimizer	Adam
Learning rate	0.001
Batch size	32
Epochs	6

Table 2: Settings for our best LSTM model

a percentage of words are randomly replaced with a special mask token. The model is subsequently required to predict the masked word. For next sentence prediction, the model predicts whether a given sentence follows from the previous or not. The idea of both techniques being that the model gains understanding of language in the process, enabling it to be fine-tuned on other tasks involving language.

Next, we experimented with both DistilBERT cased and uncased (Sanh et al., 2019). DistilBERT is a distilled version of BERT which is made to be smaller and faster than BERT, while retaining similar probabilities to the BERT base model. The model is trained on the same dataset, but only on the task of masked language modelling. Furthermore, dynamic masking was used during training, to make sure a different word is masked each epoch. The reason we chose DistilBERT is because it is more suitable to experimentation, since training times are considerably shorter. Moreover, since both models have different versions for taking casing into account, it is possible to compare the effect of casing on the current task.

We also tested RoBERTa (Liu et al., 2019), which is an improved version of BERT. RoBERTa is trained on more data, in larger batches, and with longer sequences. Similar to DistilBERT, RoBERTa is only trained using masked language modelling, and the masking was done dynamically.

ALBERT was also tried (Lan et al., 2019), which is another BERT-like model. It was trained on the same data as BERT, and in a similar fashion, using masked language modelling, as well as sentence ordering prediction, whereby the order of a pair of sentences is predicted. The distinguishing feature of ALBERT is that more of its layers share the same weights, to reduce memory consumption, and speed up training.

Subsequently, we experimented with BART (Lewis et al., 2019). Bart is different from BERT-like models in that it is an encoder-decoder model. Additionally, it was trained to reconstruct text that was arbitrarily obscured with a noising function. Despite this, the encoder side of the model can still be used for text classification.

Finally, we did an experiment with DeBERTa (He et al., 2021), which is an improved version of BERT and RoBERTA. What sets DeBERTa apart is that it features disentangled attention, meaning each word has separate vectors for content and position.

The baseline results of the models described above can be found in Table 3.

Interestingly, despite many of the aforementioned models being newer and generally considered better, the best performing model is the cased version of BERT base. Curiously, while the best BERT model is cased, the best DistilBERT model is uncased. Perhaps this suggests casing is not crucially important for the given task. Overall it seems BERT and DistilBERT indeed have very similar performance. It could therefore be argued that, because the BERT cased and DistilBERT uncased models only differ slightly in terms of loss, it is better to opt for the more efficient option. However, since our aim is optimal performance, we continue with BERT base cased going forward.

4.2.2 Fine-tuning best model

To try to optimize the best model from the previous experiments (the cased version of the BERT base model), we experimented with different learning rates, a learning rate scheduler, different batch sizes and different maximum sequence lengths for the tokenizer.

For both a batch size of 16 and 32, we first tried the different learning rates (for the Adam optimizer) of 5e-5, 3e-5 and 2e-5. According to the creators of the BERT model (Devlin et al., 2019), these values work well for a variety of fine-tuning

	val_loss	val_accuracy	epochs
bert_base_cased	0.264	92.5	5
bert_base_uncased	0.349	91.7	4
distilbert_base_cased	0.369	90.5	5
distilbert_base_uncased	0.278	92.5	5
roberta-base	0.381	90.3	5
albert-base-v2	0.387	89.0	5
bart-base	0.417	90.0	5
deberta-base	0.404	88.0	5

Table 3: Baseline results of PLMs.

experiments. We also experimented with a learning rate scheduler (polynomial decay), with an initial learning rate of 2e-5, decay steps of 1000, an end learning rate of 1e-7 and the values 0.5, 1.0 and 1.5 for power. The maximum sequence length used for these initial experiments is 100.

While the authors (Devlin et al., 2019) suggest that 2, 3 or 4 epochs work well for their experiments, we set the number of epochs to 50, and use an early stopping callback with a patience value of 5 to stop the fine-tuning process when the validation loss does not improve for five consecutive epochs. However, we found that indeed the best values were found after either 2, 3 or 4 epochs.

For a batch size of 16, we got the best results with the polynomial decay learning rate scheduler with a power value of 1.0. This gave us a validation loss of 0.191 and a validation accuracy of 93.7%. Changing the batch size to 32 did not yield better results than this.

We then tried different maximum sequence lengths for the tokenizer. Besides the initial 100, we also tried 200 and 300. The sequence length of 300 gave the best results: a validation loss of 0.157 and a validation accuracy of 95.5%. An overview of the settings of our best model can be found in Table 4.

Model	cased BERT
Batch size	16
Epochs	4
Learning rate scheduler	polynomial decay
Initial learning rate	2e-5
Decay steps	1,000
End learning rate	1e-7
Power	1.0
Max. sequence length	300

Table 4: Settings for our best pre-trained language model

5 Results

5.1 LSTM Baseline

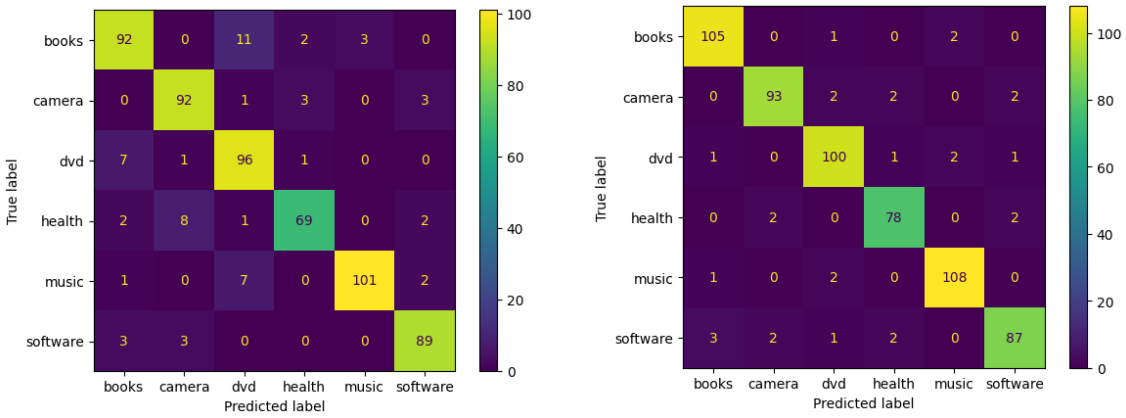
The final score of the multi-layer LSTM network on the test set was 87.7% accuracy after 16 epochs of training. From our bi-directional LSTM experiments, the best model has a loss of 0.319 on the test set and an accuracy of 89.8%. The accuracy is similar to that of the development set. The loss however, is slightly higher than on the developments set (loss on the development set was 0.296). This is a very small difference though, suggesting that the model generalizes fairly well. Since the bi-directional version scored better, the latter was chosen as our baseline score.

Table 5 shows the f1-scores per class for the bi-directional model. Here you can see that the classes are all predicted with a score of at least 0.86. The best class, music, even has an f1-score of 0.94. The confusion matrix shown in Figure 1a, shows that model mostly confuses books for DVDs, health for camera and music for DVDs.

5.2 Pre-trained language model

Our best pre-trained language model, of which the settings can be found in Table 4, resulted in a loss of 0.145 and an accuracy of 95.2% on the test set. This result is similar to the results on the validation set, meaning the model generalizes well. The loss on the test set is even slightly lower than on the development set.

The overall results and the results per class for the BERT model can be found in Table 5. It shows that on all classes the BERT model performed better than the baseline LSTM model. Only for the software class do the models have the same f1-



(a) LSTM

(b) BERT

Figure 1: Confusion matrices of our best LSTM and BERT models.

	LSTM	BERT
	f1-score	
books	0.86	**0.96**
camera	0.91	**0.95**
dvd	0.87	**0.95**
health	0.88	**0.95**
music	0.94	**0.97**
software	**0.93**	**0.93**
accuracy	0.90	**0.95**
macro f1	0.90	**0.95**
loss	0.32	**0.14**

Table 5: Results for our best LSTM and BERT models (see Table 2 and Table 4 for their details) on the test set. Best results are shown in bold.

score. While the LSTM model had a few classes that it confused, the BERT model, as can be seen in the confusion matrix in Figure 1b has no particular classes that it confuses a lot.

6 Discussion/Conclusion

Both our LSTM model and our final fine-tuned BERT model performed quite well with accuracy scores of 90% and 95% respectively. The LSTM model, however, had some trouble predicting the books, DVD, health and music classes. That the model confuses the classes book, DVD and music for each other (see the confusion matrix in Figure 1a) is not entirely unexpected since these classes are quite similar and probably contain similar words. What is unexpected is that the model confuses health for camera.

Our final BERT model performed very well on all classes. That this model would be able to make better predictions than the LSTM model was also expected since this model was already pre-trained on a lot of data. Not only did the BERT model get a better accuracy score, it also got a much lower loss. Where the LSTM model had a loss of 0.32, the BERT model had a loss of only 0.14.

Many of the experiments were done by manually changing parameter values. Using a form of grid search for all experiments would probably have resulted in even better performance. Using the Keras tuner, however, did affect the reproducibility of the results and seemed to inflate the accuracy scores. Overall, however, it is interesting to see that a simple LSTM network is able to predict quite well, even compared to a much larger and complex pre-trained language model. On the other hand, the results do show that pre-trained language models are still state-of-the-art when it comes to multiclass topic classification.

References

Qingyu Chen, Alexis Allot, Robert Leaman, Rezarta Islamaj, Jingcheng Du, Li Fang, Kai Wang, Shuo Xu, Yuefu Zhang, Parsa Bagherzadeh, Sabine Bergler, Aakash Bhatnagar, Nidhir Bhavsar, Yung-Chun Chang, Sheng-Jie Lin, Wentai Tang, Hongtong Zhang, Ilija Tavchioski, Senja Pollak, Shubo Tian, Jinfeng Zhang, Yulia Otmakhova, Antonio Jimeno Yepes, Hang Dong, Honghan Wu, Richard Dufour, Yanis Labrak, Niladri Chatterjee, Kushagri Tandon, Fréjus A A Laleye, Loïc Rakotoson, Emmanuele Chersoni, Jinghang Gu, Annemarie Friedrich, Subhash Chandra Pujari, Mariia Chizhikova, Naveen Sivadasan, Saipradeep VG, and Zhiyong Lu. 2022. Multi-label classification for

biomedical literature: an overview of the BioCreative VII LitCovid Track for COVID-19 literature topic annotations. *Database*, 2022:baac069, 08.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Sooji Ha, Daniel J Marchetto, Sameer Dharur, and Omar I Asensio. 2021. Topic classification of electric vehicle consumer experiences with transformer-based deep learning. *Patterns*, 2(2).

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Jürgen Schmidhuber, Sepp Hochreiter, et al. 1997. Long short-term memory. *Neural Comput*, 9(8):1735–1780.

Marieke van der Holt, Jarno van Houten, and Frieso Turkstra. 2023.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.